*What is claimed is:*

1. A multi-threading processor, comprising:

a first instruction fetch unit to receive a first thread and a second instruction fetch unit to receive a second thread;

an execution unit to execute said first thread and said second thread; and

a multi-thread scheduler coupled to said first instruction fetch unit, said second instruction fetch unit, and said execution unit, wherein said multi-thread scheduler is to determine whether the width of said execution unit.

2. A multi-threading processor as recited in claim 1, wherein the multi-thread scheduler unit determines whether the execution unit is to execute the first thread and the second thread in parallel depending on the width of the execution unit.

3. A multi-threading processor as recited in claim 2, wherein the multi-thread processor is an in-order processor.

4. A multi-threading processor as recited in claim 3, wherein the execution unit executes the first thread and the second thread in parallel.

5. A multi-threading processor as recited in claim 3, wherein the execution unit executes the first thread and the second thread in series.

6. A multi-threading processor as recited in claim 3, wherein the first thread and the second thread are compiled to have instruction level parallelism.

7. A multi-threading processor as recited in claim 6, further comprising:

a first instruction decode unit coupled between the first instruction fetch unit and the multi-thread scheduler; and

a second instruction decode unit coupled between the second instruction fetch unit and the multi-thread scheduler.

8. A multi-threading processor as recited in claim 4, wherein the execution unit executes only two threads in parallel.

9. A method for scheduling threads in a multi-threading processor, comprising:

determining whether said multi-threading processor is wide enough to execute a first thread and a second thread in parallel; and

executing said first thread and said second thread in parallel if said multi-threading processor is wide enough to execute the first thread and the second thread in parallel.

10. A method for scheduling threads as recited in claim 9, further comprising executing the first thread and the second thread in series if said multi-threading processor is not wide enough.

11. A method for scheduling threads as recited in claim 10, wherein the multi-threading processor is an in-order processor.

12. A method for scheduling threads as recited in claim 11, further comprising compiling the first thread and the second thread, wherein the first thread and the second thread have instruction level parallelism.

13. A method for scheduling threads as recited in claim 12, wherein the multi-threading processor executes only two threads in parallel.

14. A method for scheduling threads as recited in claim 13, further comprising:

fetching the first thread and the second thread; and

decoding the first thread and the second thread.

15. A set of instructions residing in a storage medium, said set of instructions capable of being executed by a processor for searching data stored in a mass storage device comprising:

determining whether said multi-threading processor is wide enough to execute a first thread and a second thread in parallel; and

executing said first thread and said second thread in parallel if said multi-threading processor is wide enough to execute the first thread and the second thread in parallel.

16. A method for scheduling threads as recited in claim 15, further comprising executing the first thread and the second thread in series if said multi-threading processor is not wide enough.

17. A method for scheduling threads as recited in claim 16, wherein the multi-threading processor is an in-order processor.

18. A method for scheduling threads as recited in claim 17, further comprising compiling the first thread and the second thread, wherein the first thread and the second thread have instruction level parallelism.

19. A method for scheduling threads as recited in claim 18, wherein the multi-threading processor executes only two threads in parallel.

20. A method for scheduling threads as recited in claim 19, further comprising:

fetching the first thread and the second thread; and

decoding the first thread and the second thread.